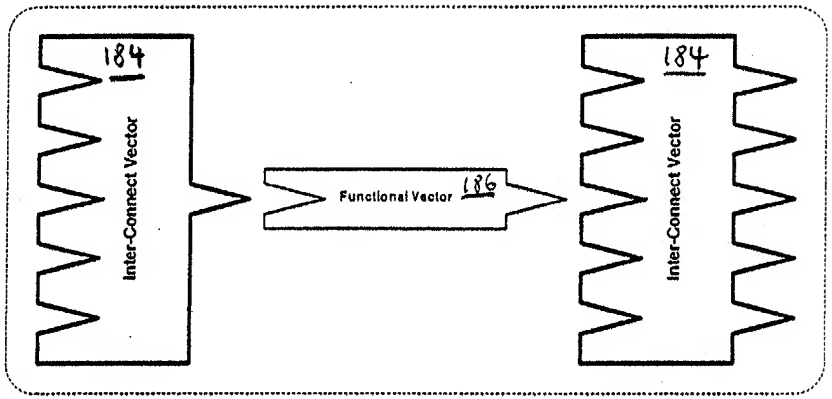

Figure 1A

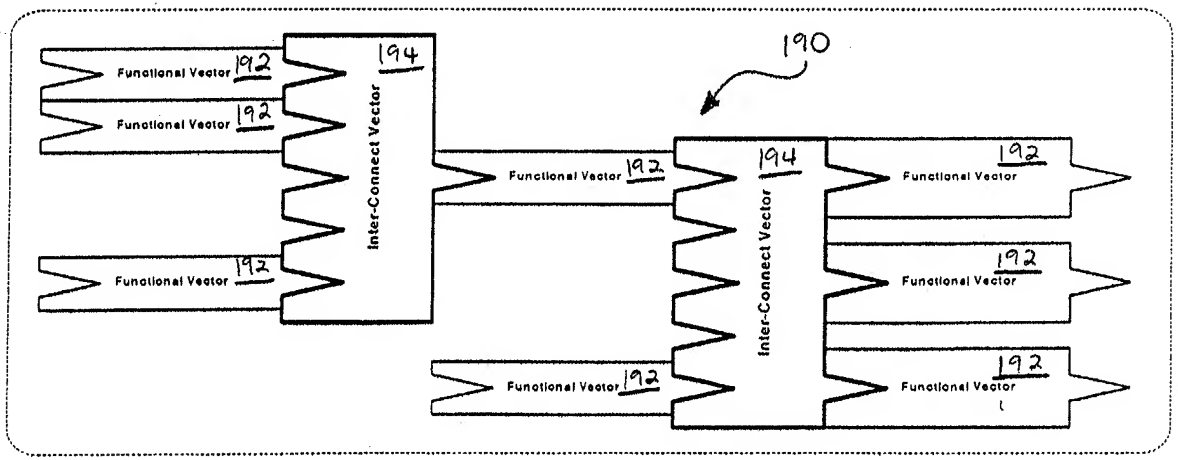

Figure 1B

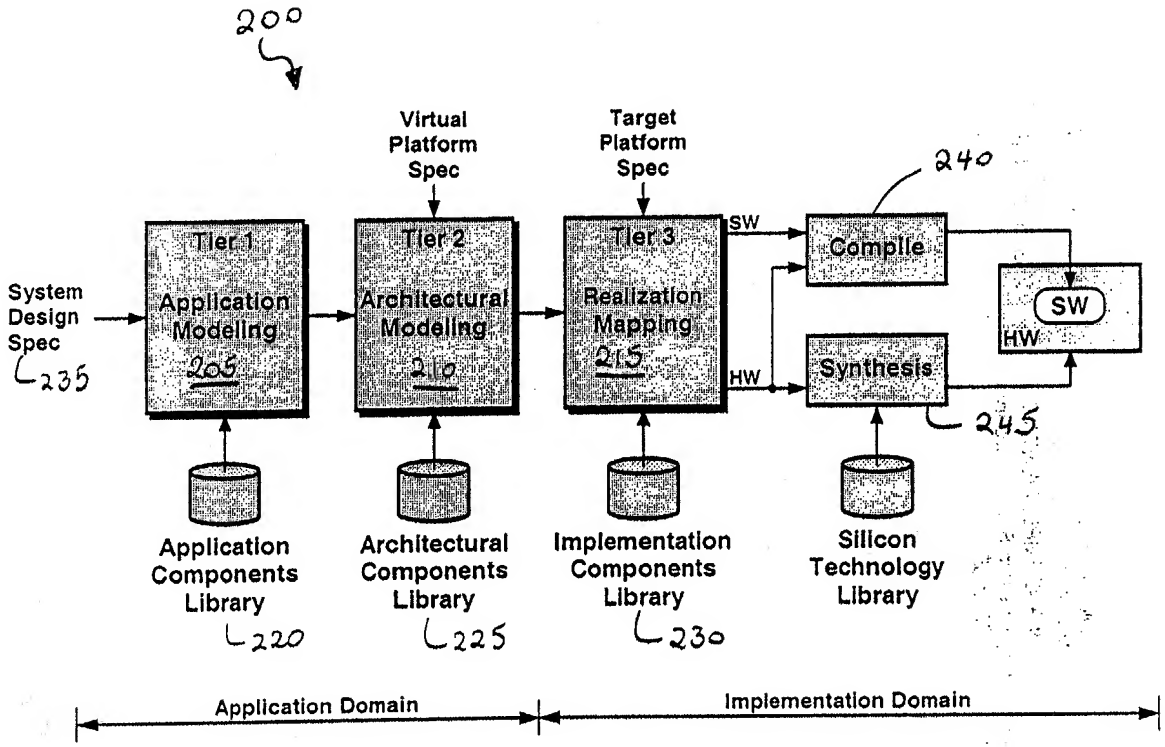

Figure 2A

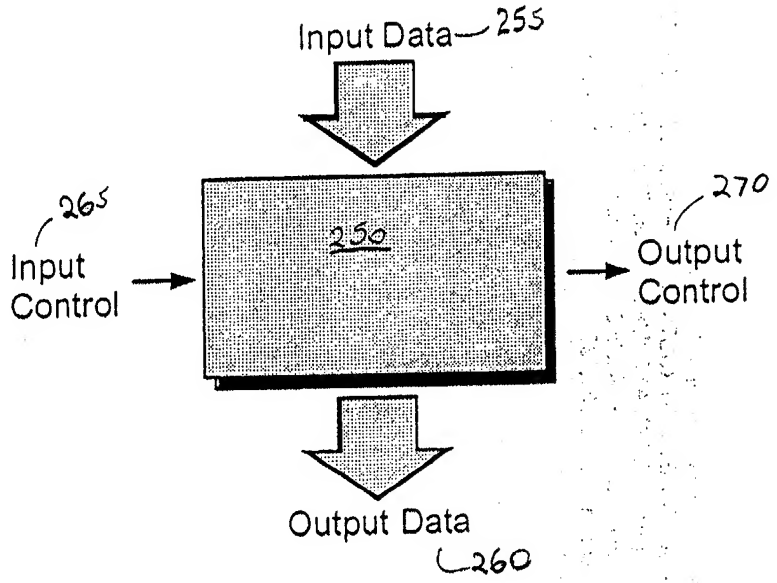
Input Data
255
265
Input
Control
250
270
Output
Control
Output Data
260

Figure 2B

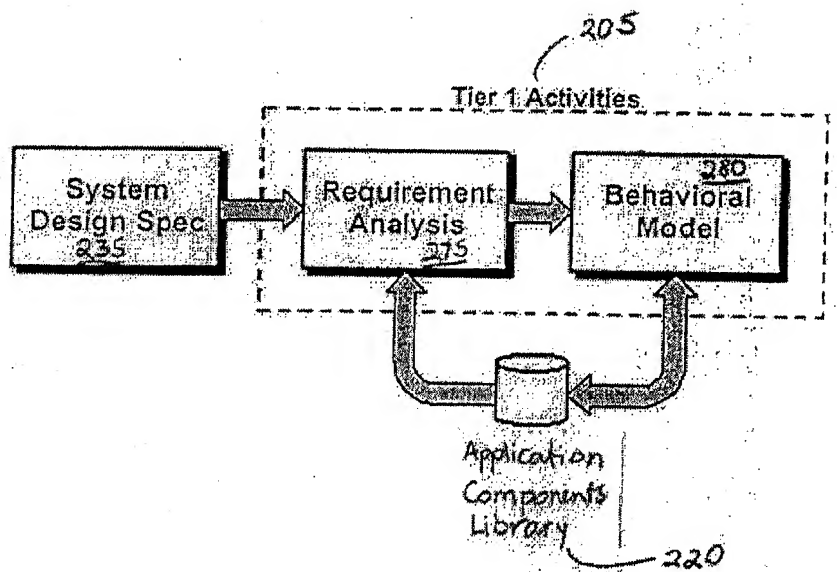
205
Tier 1 Activities
System Design Spec 235
Requirement Analysis 275
280 Behavioral Model
Application Components Library
220

Figure 2C

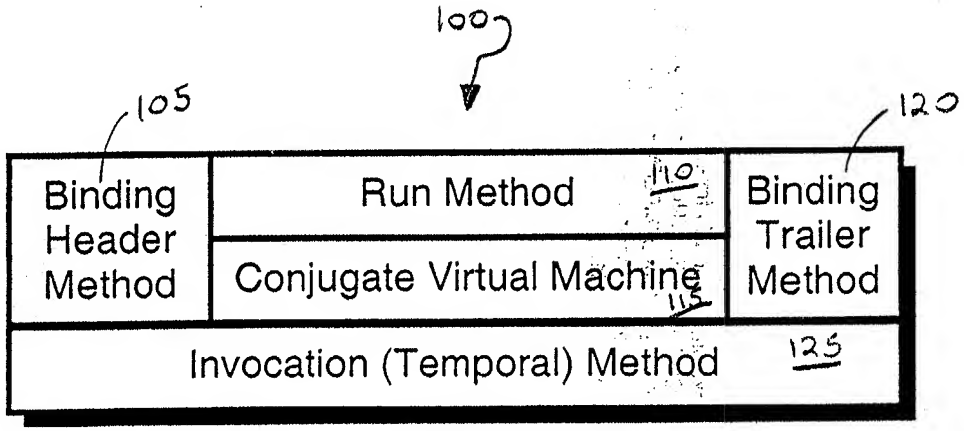
100
105
120
Binding Header Method
Run Method
110
Conjugate Virtual Machine
115
Binding Trailer Method
Invocation (Temporal) Method
125

Figure 3A

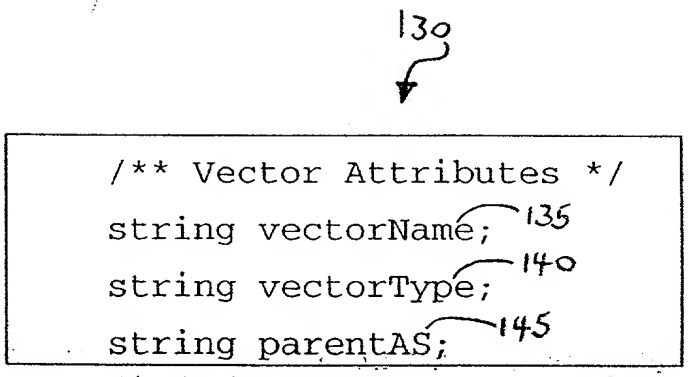

Figure 3B

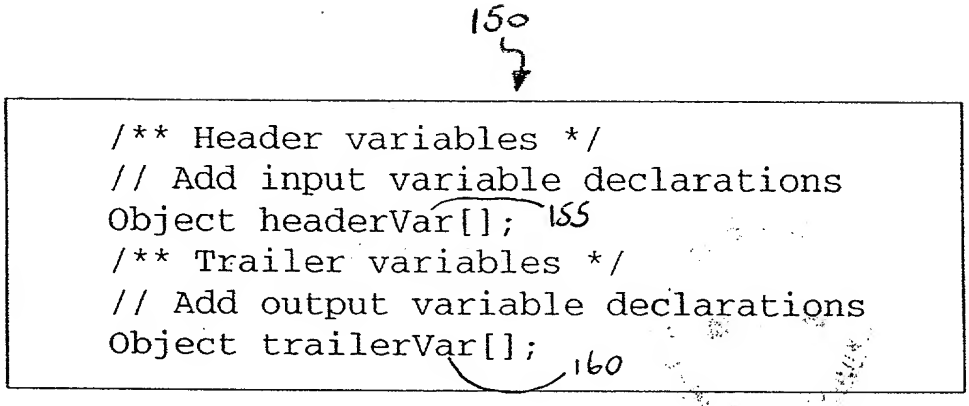

Figure 3C

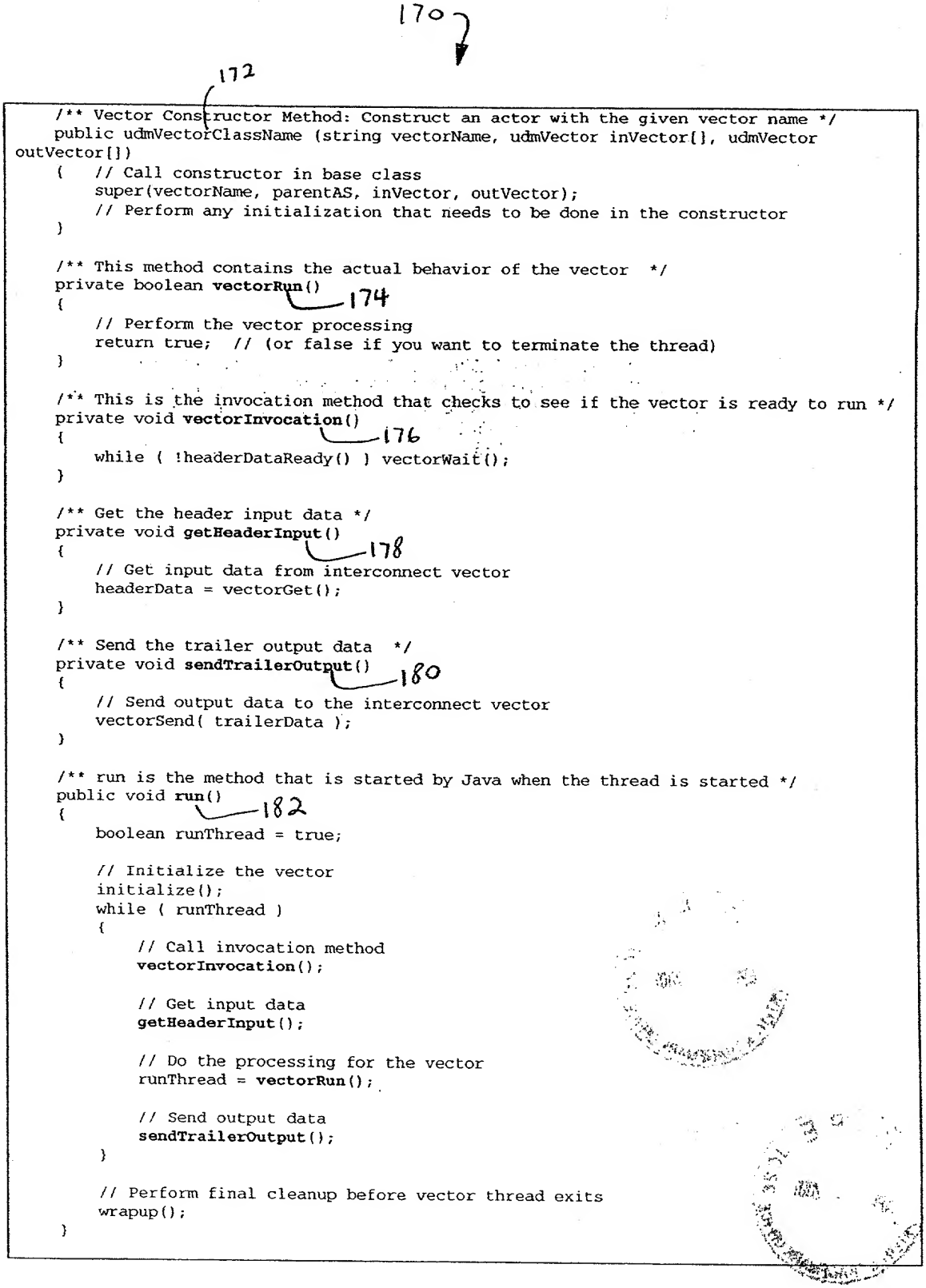
170
172
/** Vector Constructor Method: Construct an actor with the given vector name */
public udmVectorClassName (string vectorName, udmVector inVector[], udmVector outVector[])
{ // Call constructor in base class
super(vectorName, parentAS, inVector, outVector);
// Perform any initialization that needs to be done in the constructor
}
/** This method contains the actual behavior of the vector */
private boolean vectorRun()
174
{
// Perform the vector processing
return true; // (or false if you want to terminate the thread)
}
/** This is the invocation method that checks to see if the vector is ready to run */
private void vectorInvocation()
176
{
while (!headerDataReady()) vectorWait();
}
/** Get the header input data */
private void getHeaderInput()
178
{
// Get input data from interconnect vector
headerData = vectorGet();
}
/** Send the trailer output data */
private void sendTrailerOutput()
180
{
// Send output data to the interconnect vector
vectorSend(trailerData);
}
/** run is the method that is started by Java when the thread is started */
public void run()
182
{
boolean runThread = true;
// Initialize the vector
initialize();
while (runThread)
{
// Call invocation method
vectorInvocation();
// Get input data
getHeaderInput();
// Do the processing for the vector
runThread = vectorRun();
// Send output data
sendTrailerOutput();
}
// Perform final cleanup before vector thread exits
wrapup();
}

Figure 3D

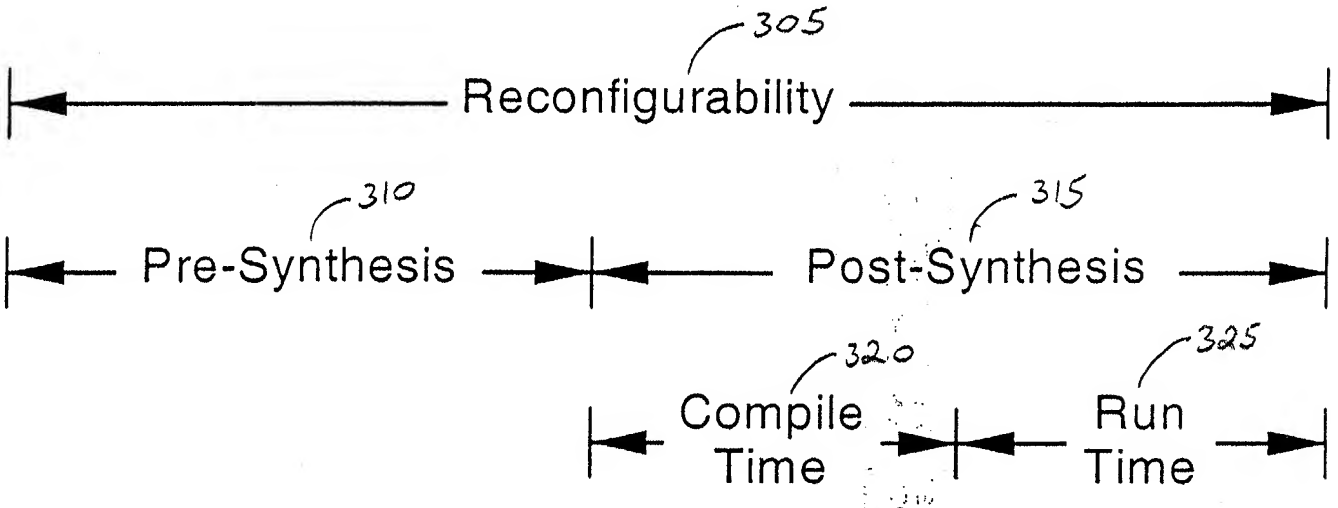
305
Reconfigurability
310
Pre-Synthesis
315
Post-Synthesis
320
Compile Time
325
Run Time

Figure 4